Final Exam (Thurs version)
APPM/MATH 4650 Fall '20 Numerical Analysis

Due date: Thur/Sat, Dec. '20, 1:30 to 4, via Gradescope/Canvas/Proctorio. **Instructor**: Prof. Becker

Instructions There are **two components** to this final. The rules are *different* than for the midterm

50 points The online Canvas "quiz" which is true/false or multiple choice.

100 points The written part (with questions listed below on this document; currently we just have place-holder questions).

Both portions of the test are taken as part of the same Canvas "quiz", as this will activate Proctorio, so do not end the quiz until you have finished *both* portions of the exam

You'll have access to your computer, and can use basic programming if needed, but you may *not* use Matlab/python's advanced features (such as their builtin root-finders or ODE solvers, for example, unless otherwise specified). You **cannot use the internet** other than for uploading to Gradescope, or checking Canvas/Piazza, or connecting to `colab` or something similar. In particular, you may **not use wikipedia or stackexchange or google websites**.

Unlike the midterms, these two components can be taken in any order, and there is a single combined time limit of 2.5 hours. Both sections are open-note, open-book (Burden and Faires 9 or 10th edition), but closed internet. If you have any notes on the cloud or use github, please download these locally before taking the exam. We'll use Proctorio which monitors your internet traffic; on Monday of finals week we'll do a dry-run of the software.

This exam only works if you follow the CU Honor Code. Violating the rules of the exam are simply not fair to your fellow students. Do not discuss any aspect of this exam with other students until after 4 PM Saturday (the two sections of the class will take slightly different exams, but still, do not discuss until after both sections have taken the exam).

Have questions? Ask on Zoom (via chat, or we can talk in a break out room).

On neither portion of the exam are you allowed to use a symbolic math program (graphing calculator, Mathematica, Maple, Desmos, Sage, Wolfram Alpha, Matlab/Python with symbolic packages, etc.). You *can* use a calculator if you want.

Problem 1: [25 points]

a) Use Newton's method to solve

$$\sin(x) = 0$$

starting at $x_0 = 3$, trying to find the root $p = \pi$. Implement this using Matlab or Python; turning in your code isn't necessary but could help with partial credit if you get the wrong answer (if your code is very short, you can also write it on paper by hand). Print out your value of x_1 to 6 digits (e.g., 1.23456), and estimate $|x_2 - \pi|$ and $|x_3 - \pi|$ to 2 digits in scientific notation (e.g., 1.1×10^{-3}).

b) Now consider rewriting the root-finding problem $\sin(x) = 0$ as a fixed-point equation, $x = g(x)$. Note that $g(x) = x + \sin(x)$ and $g(x) = x - \sin(x)$ are both valid formulations, i.e., they both have π as a fixed point. If we are to try to find the fixed point π using fixed-point iterations starting at $x_0 = 3$, is it better to use $g_+(x) = x + \sin(x)$ or is it better to use $g_-(x) = x - \sin(x)$? Explain your answer.

Problem 2: [25 points] Parts (a) and (b) are *unrelated*.

a) Consider the finite difference formula

$$\frac{-\frac{2}{5}f(x-h)+\frac{1}{2}f(x+2h)+cf(x+4h)}{h}$$

Is there a value of c for which this is an $O(h^2)$ approximation of $f'(x)$? If so, find that value of c and explain your work; if not, explain why not.

b) Consider the problem of calculating $\int_a^b f(x)\,dx$ using the nodes $\{a+h, a+2h\}$ with $h = (b-a)/3$. Derive a quadrature scheme using these nodes that has degree of accuracy (aka degree of exactness) 1 for this integral.

Problem 3: **[25 points]** Consider the IVP $y' + 10y = \sin(0.001 \cdot t)$ with $y(0) = 1$ for $t \in [0, 1000]$. If we want to find a numerical approximation of the solution to a medium level of accuracy and which can be computed quickly, which of the methods { forward Euler, backward Euler, an Adams Bashforth (AB) method, an Adams Moulton method (AM), a backward differentiation formula (BDF), an explicit Runge Kutta formula } would you use? Justify your answer. There may be more than one valid answer, so we are grading based on justification.

Problem 4: **[25 points]** Parts (a) and (b) are *unrelated*.

a) Let the $n \times n$ matrix A and the $n \times 1$ vector $\mathbf{b}$ be defined as

$$A = \begin{bmatrix} 1 & \dots & 2^{4-n} & 2^{3-n} & 2^{2-n} & 2^{1-n} \\ \ddots & & & & \vdots & \\ 0 & \dots & 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} \\ 0 & \dots & 0 & 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & \dots & 0 & 0 & 1 & \frac{1}{2} \\ 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Find the 1st component of the vector $\mathbf{x}$ that solves $A\mathbf{x} = \mathbf{b}$, clearly showing your work.

b) Explain, in your own words, how we use a pivoted LU factorization to solve a generic system of linear equations $A\mathbf{x} = \mathbf{b}$, and compare to using $\mathbf{x} = A^{-1}\mathbf{b}$ in terms of any pros and cons (e.g., compare the error, asymptotic flop count, and practical time complexity)

Problem 5: **Extra credit; no points, just "street cred"** Suppose we have an algorithm that can multiply two $n \times n$ matrices in $O(n^{2.5})$ flops rather than $O(n^3)$ of standard matrix multiplication; this isn't just hypothetical; the Coppersmith-Winograd algorithm does it in $O(n^{2.38})$ flops.

Show how to use this fast multiplication in order to solve linear systems in time $o(n^3)$; that is, something *faster* than n^3.